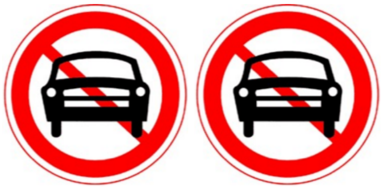
Detecting Adversarial Examples via Undercover Attack

Qifei Zhou[1],[*] Yue Wu[2], Weiping Li[1], Tong Mo[1]
[1]Peking University, [2]Nanyang Technological University
`qifeizhou@pku.edu.cn`

Abstract

1 Introduction

Both Deep nerual networks (DNNS) and Recurrent nerual networks (RNNS) are vulneralbe to adverisarial examples [18, 7, 13, 15], imperceptible modifications to the original inputs of a classifier can cause the model to produce incorrect outputs. This problem is especially important in safety-critical applications such as self-driving cars. To illustrate how adversarial samples make a system based on DNNs vulnerable, consider the following images 1:



(a) prohibition sign (b) fire sign

Figure 1. (a) is the original image of a prohibition sign and (b) is the adversary of (a)

To humans, these two images look the same: we identify each of them as a prohibition sign. The image on the left is indeed an ordinary image of a prohibition sign. We produced the image on the right by adding a very small perturbation that forces a particular DNN to classify it as a fire sign. The work or Kurakin et al. [10] showed that these transformations are effective in the physical world. Here, someone with ulterior motives can make self-driving cars behave dangerously taking advantage of the vulnerability of DNNs.

There are numerous methods have been proposed to fool the model in image classification tasks, such as L-BFGS [18], FGSM [7], BIM [10], JSMA [14] and C&W [2]. This problem has aroused great concern in the academic world. Many researchers has being trying to explain adversarial examples and find new ways to defend aganist these attack methods. There are numerous defense techniques including image compression or filtering [4, 19], defensive distillation [16] and many defenses summarized as Gradient masking [13] or Obfuscated Gradients [1]. Unfortunately, none of these defenses is yet completely satisfactory, they can generally be evaded by stronger attacks wholly or partially. The most popular defense in current research papers is probably adversarial training proposed by Goodfellow et al. [7] which also helps us a lot in *Undercover Attack*.

Due to the challenge of defenses, many recent work has turned to detect adversarial examples. Xu et al. used feature squeezing to detect, while such detectors may work well against certain attacks,

[*]Prof. Qifei Zhou is the corresponding author.

Preprint. Work in progress.

detectors in this category have been shown to fail in the white box case, where the attacker is aware of the detector [9]. Feinman et al. [6] showed that Kernel Density estimates and Bayesian Uncertainty estimates can detect points lie in low-confidence regions of the input space. Pang et al. [12] proposed a new loss function named Reverse Cross-entropy which can improve the performance of KD and BU. Howerver, KD and BU have been debeated by C&W in [3] who showed that incorporating kernel density into the objective function makes detection substantially more difficult. Ma et al. [11] proposed Local Intrinsic Dimensionality which is claimed robust to C&W and achieves quite execellent performance on various attacks both in black-box and white-box. KD, BU and LID all rely on knowledge of the attack mechanism. They need to train on adversarial examples generated by these attack methods which can't cope with new attacks. Inspired by the use of biometric and cryptograpic signatures, Dathathri et al. [5] proposed NeuralFingerprinting (NeuralFP). They trained the model with NeuralFP which achieved near perfect AUC-scores aganist black-box attacks. While, just like in cryptography, the model can't guarantee its effectiveness if the NeuralFP is exposed to us.

Most of these attacks and defenses can't directly be used to craft adversarial sequences misleading RNNS in the scenario of Natural language processing (NLP). Papernot et al. [15] adapted Jacobian based algrothim to craft adversarial sequences. He made some changes of JSMA in computer vision, because the set of legitimate word embedding is finite. Rosenberg et al. proposed to use Adversarial Training, Adversarial Signatures, Rnn Ensemble and Defense SeqGAN to make an RNN classifier robust aganist adversarial examples. Adversarial Training get the best detection performace with 89.39% Recall and 92.10% Accuracy.

We have been working hard to study and explain the vulnerability of normal samples all the times. However, we didn't realize that adversarial samples are more vulnerable than normal ones, especially in Computer Vision. It may be not easy to sucessfully attack a benign example, and we even can't find any adversarial examples within the ϵ norm ball of the benign example as long as ϵ is small enough. Nevertheless, there must be some small modifications which can attack an adversarial example sucessfully. Because the adversarial example is generated from a benign example, we can find at least one way which is rolling back to the benign one. This is a successful attack for the adversarial example (In our paper, a successful attack is an attack which changes the model's prediction, not necessary the true lable. Especially for an adversarial input, its prediction is not exactly the true class). In fact, our experiments show that adversarial examples is far more vulnerable than normal examples. We design *Undercover Attack* to detect adversarial examples based on this property. Althogh, the characteristics may be completely different in NLP, *Undercover Attack* can also get perfect 100% AUC-socres with no adversarial training or anything else. There are some intriging things between CV and NLP when adapted *Undercover Attack* to them, we will discuss it in detail in Section 3.

Our key contributions are:

- We propose *Undercover Attack* to detect adversarial examples. *Undercover Attack* is a novel idea which defends by attacks. We discuss how *Undercover Attack* can distinguish adversarial samples in Computer Vision and Natural language processing seperately. Experients show that it is easier to attack adversarial samples successfully than normal samples.
- *Undercover Attack* does not rely on knowledge of the attack mechanism. Our framework does not require additional detectors, networks or parameters. And we emperically show that the performance of *Undercover Attack* is robust to unknown attack methods.
- To the best of our knowledge, we are the first to introduce a adversarial detection mechanism which can be both adapted to attacks on images and sequences with excellent performance. When faced with unknown and white-box attacks, *Undercover Attack* achieves state-of-the-art AUC-scores with an average performace of 97% on various attacks on both MNIST and CIFAR10 datasets. The most exciting thing is that *Undercover Attack* achieves perfect 100% AUC-scores with RNNS on the movie review dataset IMDB.

2 Adversarial Attacks

The typical goal of an adversary is to generate a sample which looks like a normal example, but it is misclassified by the target model. It means to modify few pixels with small perturbations of inputs in CV and change few words of inputs in NLP. This amounts to making minor changes to the inputs, causes the target model to misclassify the sample, but remains correctly classify by human. A significant number of adversarial attacks satisfying this goal have been proposed in recent years. This allows us a wide range of attacks to choose from in our investigation. Here, we introduce some of the most well-known attacks.

2.1 Fast Gradient Sign Method (FGSM)

Goodfellow et al. hypothesized that adversarial examples can be found using only a linear approximation of the target model [7]. They introduced the Fast Gradient Sign Method for efficiently crafting adversarial examples.

$$X^{adv} = X + \epsilon sign(\bigtriangledown_X J(\theta, X, y_{true})) \tag{1}$$

FGSM works by linearizing loss function in L_∞ neighbourhood of a clean image. ϵ is a parameter determines the perturbation size in the direction of the gradient, θ represents the parameters of the model, y_true is the correct label of X.

2.2 Basic Iterative Method (BIM, I-FGSM)

Kurakin et al. apply FGSM multiple times with small step size [10] which called Basic Iterative Method or Iterative FGSM. α is the perturbation size in each iteration, $Clip_{X,\epsilon}(\cdot)$ performs per-pixel clipping of the image X, so X^{adv} will be in L_∞ ϵ-neighbourbood of the clean image X.

$$X_0^{adv} = X, \qquad X_{n+1}^{adv} = Clip_{X,\epsilon}\{X_n^{adv} + \alpha sign(\bigtriangledown_X J(\theta, X_n^{adv}, y_{true}))\} \tag{2}$$

2.3 Jacobian-based Saliency Map Attack (JSMA)

Papernot et al. [14] introduced Jacobian-based Saliency Map Attack for targeted misclassification. The above two methods (FGSM, BIM) modifies each pixel with small ϵ-ball perturbation, JSMA works by modifying a limited number of input pixels with relatively large perturbations instead. JSMA iteratively perturbs pixels that have high adversarial saliency scores. Euqation 3 explains how to calculate adversarial saliency scores. To achieve a target class t, $F_t(X)$ must be increased while the probabilities of all other classes $\Sigma_{j \neq t} \frac{\partial F_j(X)}{\partial X_i}$ decrease.

$$S(X,t)[i] = \begin{cases} 0, \text{ if } \frac{\partial F_t(X)}{\partial X_i} < 0 \text{ or } \Sigma_{j \neq t} \frac{\partial F_j(X)}{\partial X_i} > 0 \\ (\frac{\partial F_t(X)}{\partial X_i})|\Sigma_{j \neq t} \frac{\partial F_j(X)}{\partial X_i}|, \text{ otherwise} \end{cases} \tag{3}$$

2.4 Carlini and Wagner Attack (C&W)

N.Carlini and D.Wagner introduced a optimization attack framework [3] that passed a range of attacks. They designed a loss function that has smaller values on adversarial samples and higher on benign samples. This results in three kind of attacks: an L_∞ attack, an L_0 attack, and an L_2 attack. They achieved the strongest L_2 attack which we will consider in this paper with the following loss function:

$$\text{minimize } ||\frac{1}{2}(tanh(w) + 1) - x||_2^2 + c \cdot f(\frac{1}{2}(tanh(w) + 1)) \tag{4}$$

$f(\cdot)$ is based on the best objective function found earlier, which is defined as:

$$f(x^{'}) = max(max\{Z(x^{'})_i : i \neq t\} - Z(x^{'})_t, -k) \tag{5}$$

Here, k is a parameter control the misclassification confidence, we can find adversarial examples with high confidence by adjusting k.

2.5 Jacobian-based Attack On RNNs

Papernot et al. [15] showed that the algrothims introduced previously to craft adversarial instances misclassified by feed-forward neural networks can be adapted to recurrent neural works. However, due to the finite set of word embeddings, these methods can't be adapted to natural language processing tasks directly. That's to say that we can't modify the word embeddings casually, because we may not find a real word corresponding to the modified word embeddings. They followed a heuristic procedure to solve the problem. They iteratively find a word $\vec{z}$ in words set that the sign of the difference between the embeddings of $\vec{z}$ and the original input word is closet to $sgn(J_f(\vec{x})[i, f(\vec{x})])$. Eventually, they achieved an error rate of 100% by changing 9.18 words in 71.06-word-long sentences on average. Later on, we will discuss our method based on Jacobian which performs better in the next section.

3 Undercover Attack

Armed with introduction of adversarial attacks on DNNs and RNNs, we now introduce *Undercover Attack* aims to distinguish adversarial examples from normal ones. We notice that most defensive methods can be easily evaded by specific attacks, howerver, attack methods can always succeed with much less effort. That's to say, defense is much harder. Our detection method was inspired this characteristic, We utilize this feature to design *Undercover Attack*. If you want to find an adversarial sample which can bypass *Undercover Attack*, you need to take two steps: First, you need to craft an adversarial sample that fool the model successfully. Second, you should make the adversarial sample be robust to our specific attack mechanism which means if we modify the adversarial sample slightly, the predicted class of the sample won't change. For example, there is a image of cat, and you modify some pixels of the image (we get adv sample1) which successfully fool the model to recognize it as a plane. You need to guarantee that if we change some pixels of adv sample1 (we get adv sample2), the model can still classify it as a plane, not a cat or anything else. Adv sample1 is not easy to find, because we need to make it be robust to a fake class. That's to say, we need to find an adversarial sample always classified as a same fake class by the target model when facing new attacks. This's theoretically impossiable, there must be at least one original sample which adv sample1 crafted from can deny this hypothesis. And we will prove that adversarial examples is in fact more vulnerable than normal ones when feed to DNNs and RNNs.

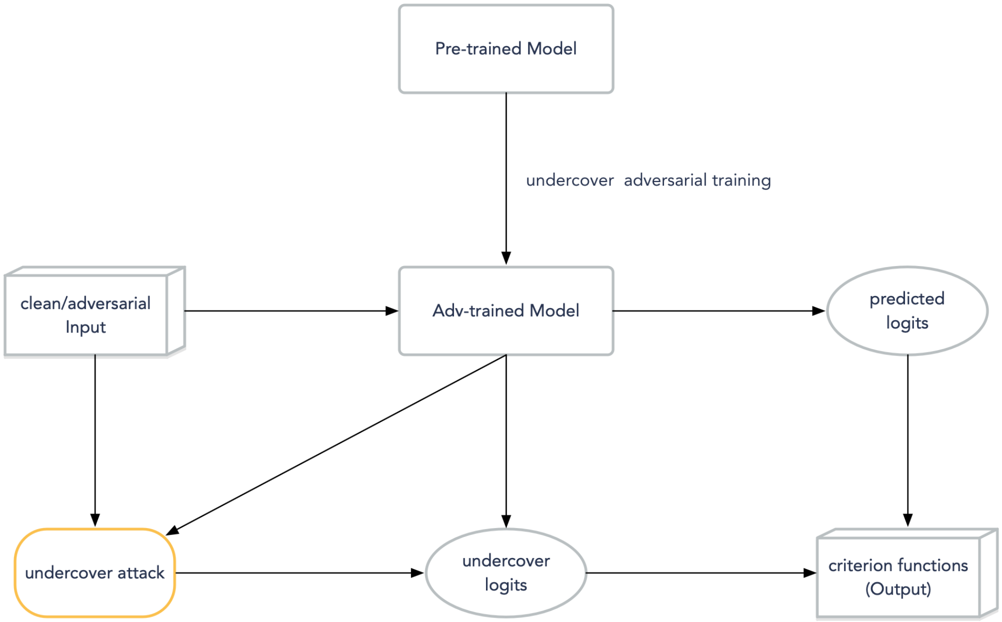
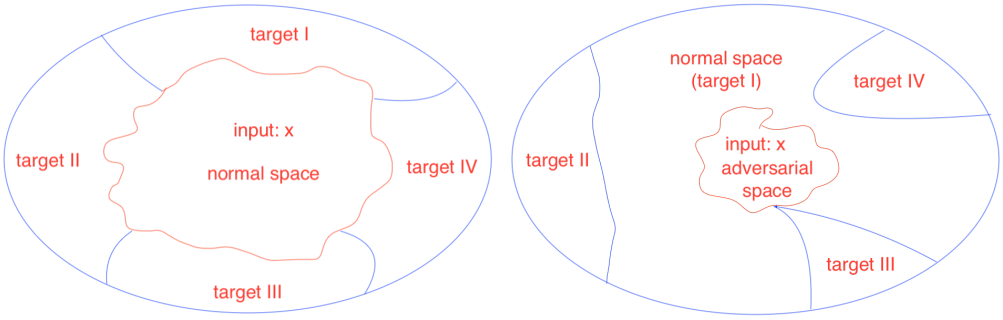


Figure 2. The framework of undercover attack on DNNs and RNNs.

3.1 Vulnerability Of Adversarial Examples

In this section, we will analyze the the vulnerability both empirically and experomentally. For an effective model, we believe that it perform correctly on most of examples. This indicate that the normal space is larger than any other target adversarial space. Our analysis is based on the hypothesis and our experiments proved this view to some extent. The model input x must be an normal input or an adversarial input as is shown in 3.



(a) normal input

(b) adversarial input

Figure 3. (a): The input is an normal example. (b): The input is an adversarial example.

Figure 3(a) represents the input is in normal space and it needs relatively large modification to become an adversarial sample. Figure 3(b) shows that the input is an adversarial sample, the adversarial space is much smaller than the normal one, so it won't take too much effort to modify the input back to the normal space (target I) or the other target adversarial space (target II, III or IV).

3.2 Criteria for identification of adversaries

To quantify the vulnerability of the sample, we apply an undercover attack to the model and measure the difference between the outputs of the original example and modified example. To illustrate, if we get an example A and want to know whether it is an adversarial example or not. First, We feed A to the target model and get the output logits (logits I). Then, we apply undercover attack to A and get A^*. The next, we feed A^* to the target model to get the output logits (logits II). In the last, we caculate the difference between logist I and logits II to judge if A is an adversarial sample. Here, the logits are the probabilities of each class which are the softmax outputs of the last dense layer.

$$S_i = \frac{e^i}{\sum\limits_{j=1}^{C} e^j} \tag{6}$$

We design two criterion functions to represent the difference between logit I and logit II.

$$\begin{aligned} &t = onehot(argmax(L)) \\ &f_1(L, L^{'}) = \sum_{i=1}^{C} t(i) log(L_i^{'}) \\ &f_2(L, L^{'}) = \sum_{i=1}^{C} (L_i - L_i^{'})^2 \end{aligned} \tag{7}$$

Here, L is logits I, $L^{'}$ is logits II, t is the one-hot code of the predicted class of A, $f_1(\cdot)$ is the negative Cross Entropy of t, and $f_2(\cdot)$ is the Euclidean Distance between L and $L^{'}$. We can choose either $f_1(\cdot)$ or $f_2(\cdot)$ as the creterion function. $f_2(\cdot)$ reprsents the difference of the whole distribution of L and $L^{'}$, while $f_1(\cdot)$ focus on the change of the main class. The two criterions aim at detection of different types of adversaries. In most of our experiments, $f_1(\cdot)$ performs a little bit better.

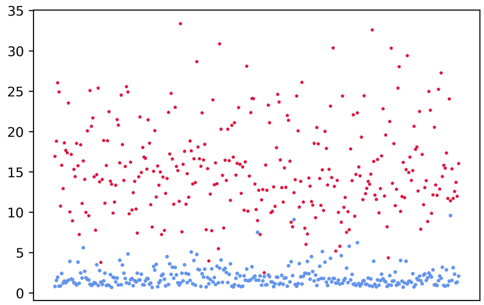


Figure 4. The framework of undercover attack on DNNs and RNNs.

4 Experiments

To verify the effectiveness of *Undercover Attack*, we present results on the detectability of adversarial perturbations on some famous datasets: MNIST, CIFAR10, ImageNet in CV and IMDB in NLP.

4.1 Network Setup

Here, we briefly describe the models used in the above classification tasks. We use a 5-layer convolutional network which achieves 99.2% accuracy on MNIST. As for CIFAR10, we use the 18-layer PreActResNet [8] as classifier. During training, the learning rate is set to 0.1 initially and decrease every 150 epochs, the total epoch is 450. The model reports 95.49% accuracy on normal samples which is comparable to the state-of-art results. On ImageNet, we use the pretrained InceptionV4 [17] which claimed 80.062% top-1 accuracy.

4.2 Performance of Undercover Attack On Image classification Tasks

no-target

Dataset	Attack Methods	L2	UK Normal Acc	UK Adv Acc	Detection ROC
MINIST $\epsilon = 0.15$	FGSM	6.16	93.50%	0.14%	99.98%
	BIM-A	2.29	93.50%	14.77%	97.48%
	BIM-B	2.96	93.50%	15.94%	97.89%
	JSMA	5.88	93.50%	13.97%	91.88%
	C&W	1.95	93.50%	4.47%	95.68%
CIFAR10 $\epsilon = 0.031$	FGSM	3.43	95.71%	5.79%	99.90%
	BIM-A	0.36	95.71%	23.84%	95.43%
	BIM-B	1.12	95.71%	13.96%	98.52%
	JSMA	7.50	95.71%	16.58%	97.67%
	C&W	0.13	95.71%	29.91%	93.49%
ImageNet $\epsilon = 0.031$	FGSM	A	A	A	A
	BIM-A	A	A	A	A
	BIM-B	A	A	A	A

Table 1. Undercover attack details

4.3 Intrinsic Properties Of Adverisarial Examples

4.4 Undercover Attack On RNNs

We use bidirectional LSTM that achieves 88.24% accuracy on IMDB dataset.

5 Conclusion

Acknowledgments

We would like to thank Zhilong Hong and Haochen Li for helpful discussions and their useful advice on the topic.

References

[1] Anish Athalye, Nicholas Carlini, and David A Wagner. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. *international conference on machine learning*, pages 274–283, 2018.

[2] Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. 2016.

[3] Nicholas Carlini and David A Wagner. Adversarial examples are not easily detected: Bypassing ten detection methods. *arXiv: Learning*, pages 3–14, 2017.

[4] Nilaksh Das, Madhuri Shanbhogue, Shangtse Chen, Fred Hohman, Li Chen, Michael E Kounavis, and Duen Horng Chau. Keeping the bad guys out: Protecting and vaccinating deep learning with jpeg compression. *arXiv: Computer Vision and Pattern Recognition*, 2017.

[5] Sumanth Dathathri, Stephan Zheng, Richard M Murray, and Yisong Yue. Detecting adversarial examples via neural fingerprinting. *arXiv: Learning*, 2018.

[6] Reuben Feinman, Ryan R Curtin, Saurabh Shintre, and Andrew B Gardner. Detecting adversarial samples from artifacts. *arXiv: Machine Learning*, 2017.

[7] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *Computer Science*, 2014.

[8] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. *european conference on computer vision*, pages 630–645, 2016.

[9] Warren He, James Wei, Xinyun Chen, Nicholas Carlini, and Dawn Song. Adversarial example defenses: Ensembles of weak defenses are not strong. 2017.

[10] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. Adversarial examples in the physical world. 2016.

[11] Xingjun Ma, Bo Li, Yisen Wang, Sarah M Erfani, Sudanthi N R Wijewickrema, Grant Schoenebeck, Michael E Houle, Dawn Song, and James Bailey. Characterizing adversarial subspaces using local intrinsic dimensionality. *international conference on learning representations*, 2018.

[12] Tianyu Pang, Chao Du, Yinpeng Dong, and Jun Zhu. Towards robust detection of adversarial examples. *neural information processing systems*, pages 4584–4594, 2018.

[13] Nicolas Papernot, Patrick Mcdaniel, and Ian Goodfellow. Transferability in machine learning: from phenomena to black-box attacks using adversarial samples. 2016.

[14] Nicolas Papernot, Patrick D Mcdaniel, Somesh Jha, Matthew Fredrikson, Z Berkay Celik, and Ananthram Swami. The limitations of deep learning in adversarial settings. *ieee european symposium on security and privacy*, pages 372–387, 2016.

[15] Nicolas Papernot, Patrick D Mcdaniel, Ananthram Swami, and Richard E Harang. Crafting adversarial input sequences for recurrent neural networks. *military communications conference*, pages 49–54, 2016.

[16] Nicolas Papernot, Patrick D Mcdaniel, Xi Wu, Somesh Jha, and Ananthram Swami. Distillation as a defense to adversarial perturbations against deep neural networks. *ieee symposium on security and privacy*, pages 582–597, 2016.

[17] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alex Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. 2016.

[18] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *Computer Science*, 2013.

[19] Weilin Xu, David Evans, and Yanjun Qi. Feature squeezing: Detecting adversarial examples in deep neural networks. *network and distributed system security symposium*, 2018.